



Universidad del Istmo de Guatemala
Facultad de Ingenieria
Ing. en Sistemas
Informatica II
Prof. Ernesto Rodriguez - erodriguez@unis.edu.gt

Laboratorio #11

Fecha de entrega: 25 de Abril, 2019 - 11:59pm

Instrucciones: Resolver cada uno de los ejercicios siguiendo sus respectivas instrucciones. El trabajo debe ser entregado a traves de Github, en su repositorio del curso, colocado en una carpeta llamada "Laboratorio #11". Al menos que la pregunta indique diferente, todas las respuestas a preguntas escritas deben presentarse en un documento formato pdf, el cual haya sido generado mediante Latex. Este laboratorio debe ser elaborado en parejas.

Tarea #1 (50%)

La función *fold* es una función universal que permite replicar cualquier transformación de una colección de elementos de forma general. Su tipo es el siguiente:

$$\forall T \forall S . S \text{ fold}(\text{std} :: \text{funcion}(S(S,T)), S, \text{std} :: \text{vector}(T))$$

Essencialmente, esta función toma como parametros una colección, de valores, un valor inicial y una función que le permite combinar un valor del mismo tipo que el valor inicial y un valor de la colección y producir un valor del mismo tipo que el valor inicial. Este proceso se repite hasta haber consumido todos los valores de la colección y se haya creado el producto final. Un ejemplo de su utilización es el siguiente:

```
int main() {
    std::vector<int> valores{1,2,3,4,5};
    std::function<int(int, int)> sumar = [](int x, int y){ return x + y; };

    // Imprime 15 (osea 1+2+3+4+5)
    printf("El resultado es %d\n", fold<int, int>(sumar, 0, valores));
}
```

Tarea #2 (50%)

La función *existe* es una función que permite buscar si un objeto existe dentro de una colección. Su tipo es el siguiente:

$$\forall T . \text{bool existe}(\text{std} :: \text{function}(\text{bool}(T)), \text{std} :: \text{vector}(T))$$

Esta función acepta un validador y una colección de objetos y aplica el validador en cada uno de los objetos. Si alguno de esos objetos pasa la validación, la función retorna *true*, de lo contrario retorna *false*. A continuación se muestra un ejemplo de su utilización:

```

int main(){
    std::vector<int> valores { 3,5,7,9 };
    std::function<bool(int)> esPar = [](int valor){ return valor % 2 == 0; };
    std::function<bool(int)> esImpar = [](int valor){ return valor % 3 == 0; };

5
    if(!find<int>(esPar, valores)){
        printf("No hay numeros pares\n");
    }

10
    if(find<int>(esImpar, valores)){
        printf("Si hay numeros impares\n");
    }
}

```

Su tarea es utilizar la función *fold* para implementar la función *existe*.

Extra #1 (+2 puntos neto)

En C++ es posible utilizar cualquier objeto que tenga un metodo llamado *operator()* como si fuese función. Por ejemplo:

```

class Sumar{
    int operator() (int a, int b){
        return a + b;
    }
5
};

int main(){
    Sumar sumar;
    printf("La suma de 1 + 2 es %d\n", sumar(1,2));

10
}

```

Sin embargo, la función *fold*, tal como fue implementada en la tarea #1 tiene la limitante que solo puede ser utilizada con instancias de `std::function`. En otras palabras, el siguiente codigo no compila:

```

class Sumar{
    int operator() (int a, int b){
        return a + b;
    }
5
};

int main(){
    std::vector<int> valores{1,2,3,4,5};
    Sumar sumar;

10
    // Imprime 15 (osea 1+2+3+4+5)
    printf("El resultado es %d\n", fold<int, int>(sumar, 0, valores));
}

```

Debido a que *fold* ha sido llamado con una instancia de *Sumar* en vez de una instancia de `std::function`. Su tarea es crear una version de *fold* que pueda funcionar con objetos de **cualquier** tipo siempre y cuando se puedan operar mediante el *operator()*.

Extra #2 (+2 puntos neto)

Cuando se trabajan proyectos grandes en C++, se acostumbra a automatizar el proceso de compilación debido a que el número de archivos involucrados suele cambiar constantemente. Una herramienta para facilitar el manejo de proyectos grandes es cmake. Su tarea es configurar un archivo CMake para compilar un programa de C++ que utilice el código de esta hoja de trabajo. Como entrega, debe incluir el archivo “CMakeLists.txt” en su entrega con las instrucciones para compilar un ejecutable. Se puede apoyar de <https://cmake.org/examples/> y <https://cmake.org/cmake/help/v3.14/manual/cmake-buildsystem.7.html#introduction> para información sobre como utilizar cmake.